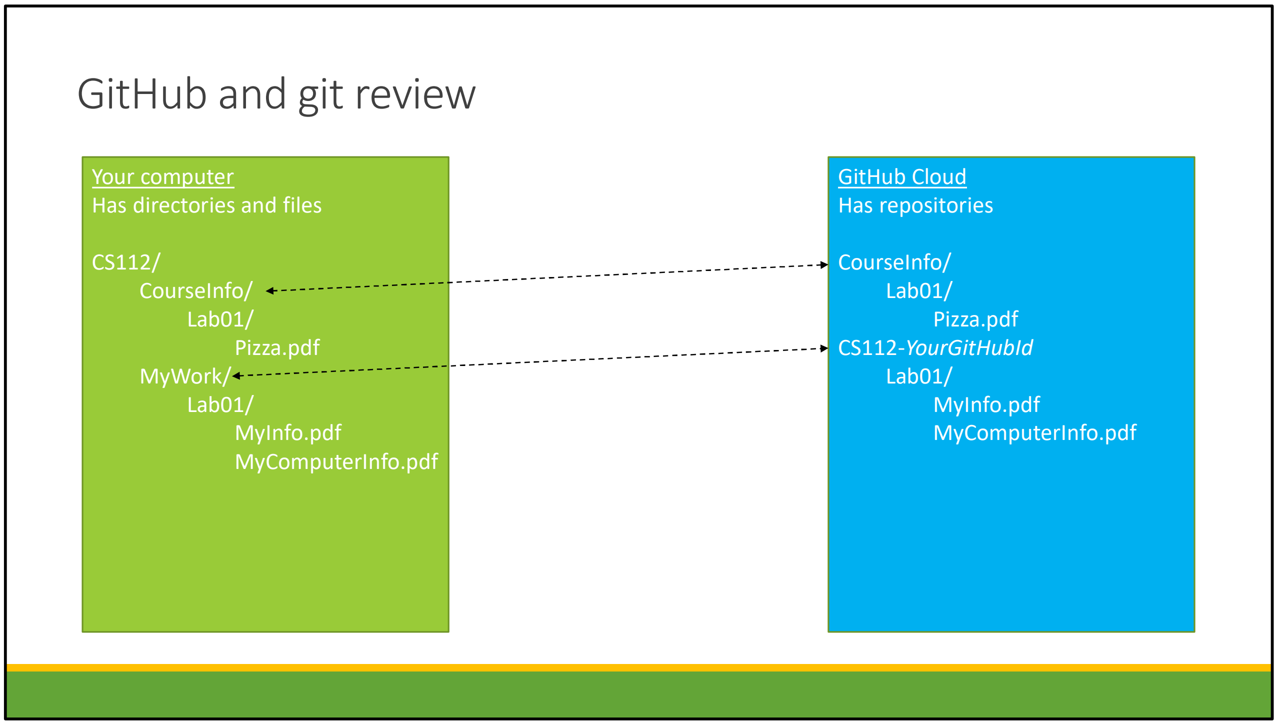
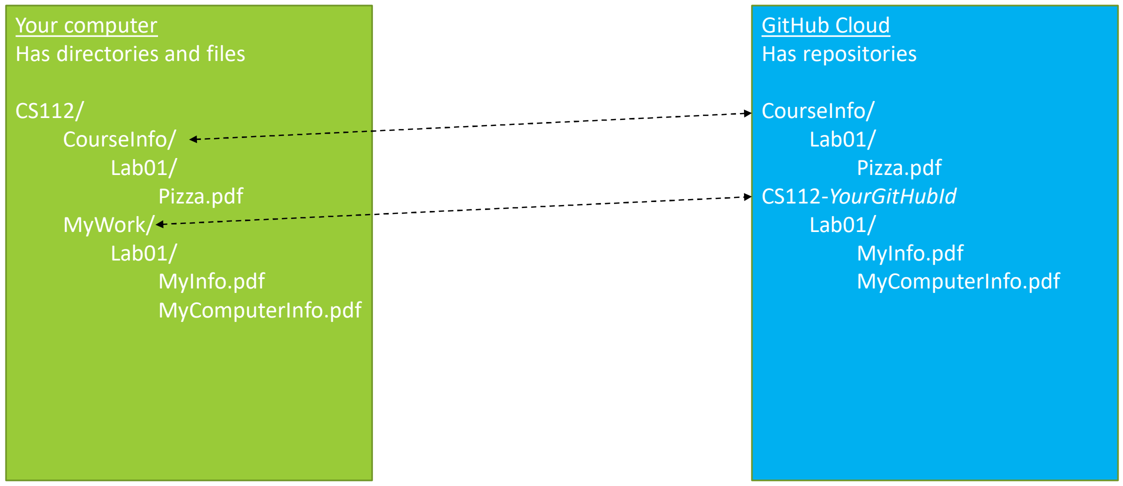
Java Beginnings

CS112 – Java Programming

August 2022



GitHub and git review



Your computer and GitHub do not stay in sync automatically.
You use "git" commands to "pull" files from GitHub to your computer.
You use "git add", "git commit", and "git push" to "push" files from your computer to the GitHub cloud.
REPOSITORIES contain current and past versions of files, along with commit comments.

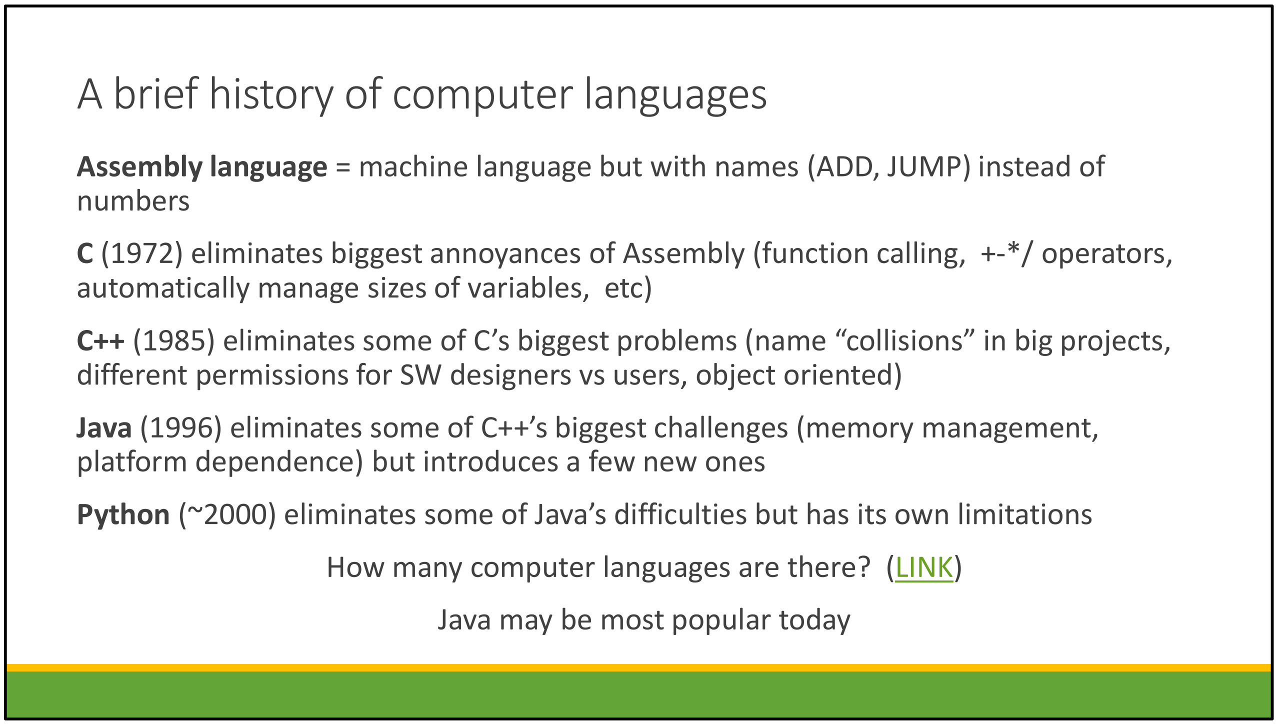
Details matter to computers!

If I ask for a directory to be called "Lab01"…

- lab01 is not the same
- Lab1 is not the same

Why does it matter?

A. Because Paul is annoying
B. Because Paul's automated grading software only knows to look for "Lab01"
C. Because if your SW must work with other people's software (or with existing libraries), it is important that your SW **match exactly** with the **specification** of the other software or libraries
D. All of the above



A brief history of computer languages

Assembly language = machine language but with names (ADD, JUMP) instead of numbers

C (1972) eliminates biggest annoyances of Assembly (function calling, +-*/ operators, automatically manage sizes of variables, etc)

C++ (1985) eliminates some of C's biggest problems (name "collisions" in big projects, different permissions for SW designers vs users, object oriented)

Java (1996) eliminates some of C++'s biggest challenges (memory management, platform dependence) but introduces a few new ones

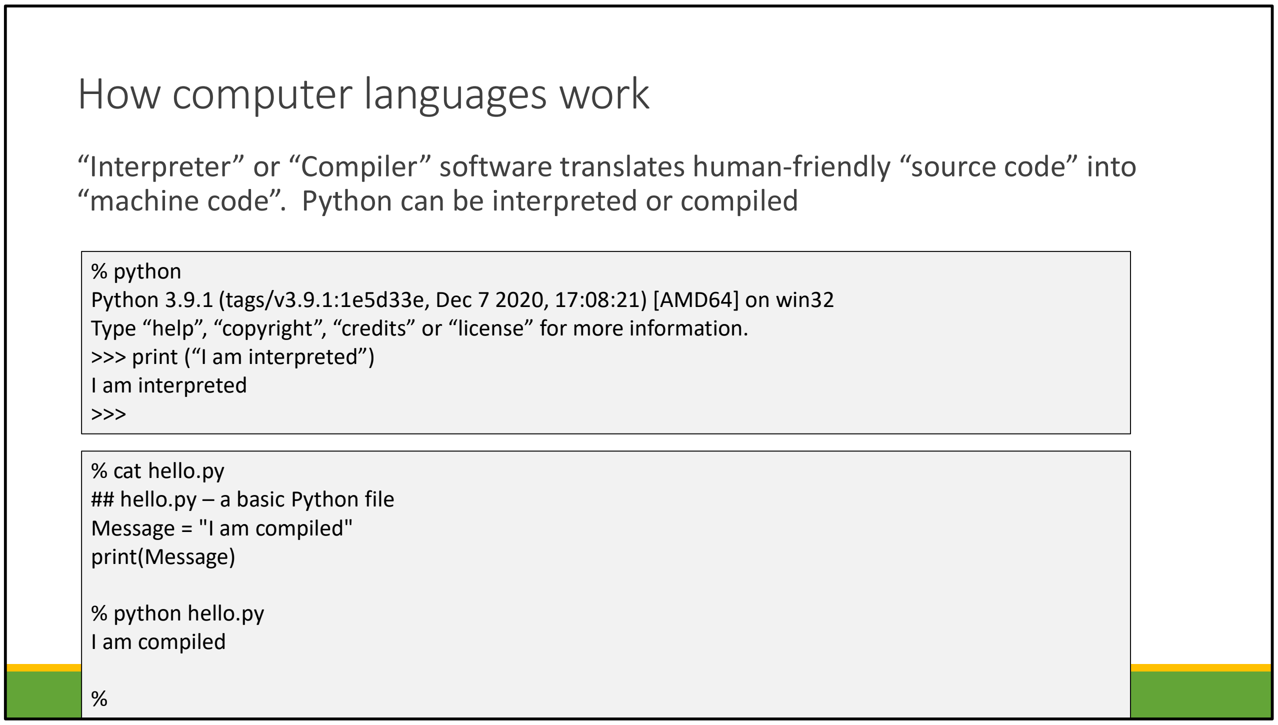
Python (~2000) eliminates some of Java's difficulties but has its own limitations

How many computer languages are there? (LINK)

Java may be most popular today

From last time: Machine Language = numbers in memory that the CPU interprets as ADD, MULT, LOAD, etc instructions

# of languages estimated to be between 300 and 9000
At my previous work, C++ and Java mostly. Some C#



How computer languages work

“Interpreter” or “Compiler” software translates human-friendly “source code” into “machine code”. Python can be interpreted or compiled

```
% python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [AMD64] on win32
Type “help”, “copyright”, “credits” or “license” for more information.
>>> print (“I am interpreted”)
I am interpreted
>>>
```

```
% cat hello.py
## hello.py – a basic Python file
Message = "I am compiled"
print(Message)

% python hello.py
I am compiled

%
```

EXPLAIN DIFFERENCE: Interpreter = LINE BY LINE execution. Compiler = Process ENTIRE FILE OF SOFTWARE

Java is compiled…

Source code files have suffix ".java"

Compiler is called "javac"

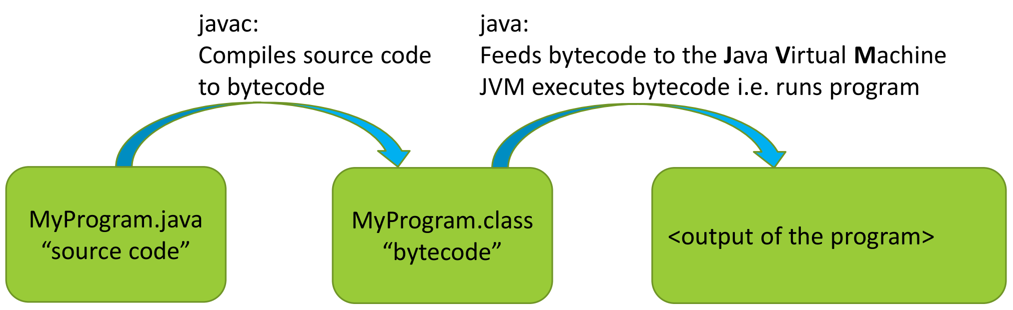
But Java does not compile to machine code! It compiles to a format called "bytecode" that must be executed by another program called "java".

```
% javac MyProgram.java

% java MyProgram
I ran my first Java program!

%
```

Java is compiled and then interpreted

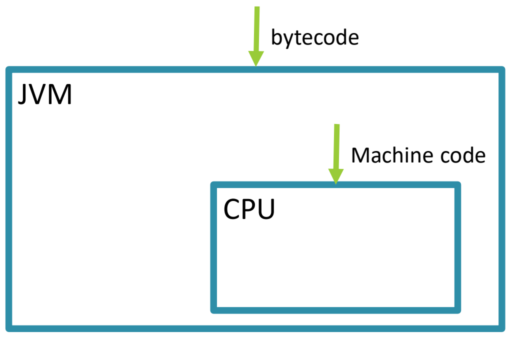

javac:
Compiles source code
to bytecode

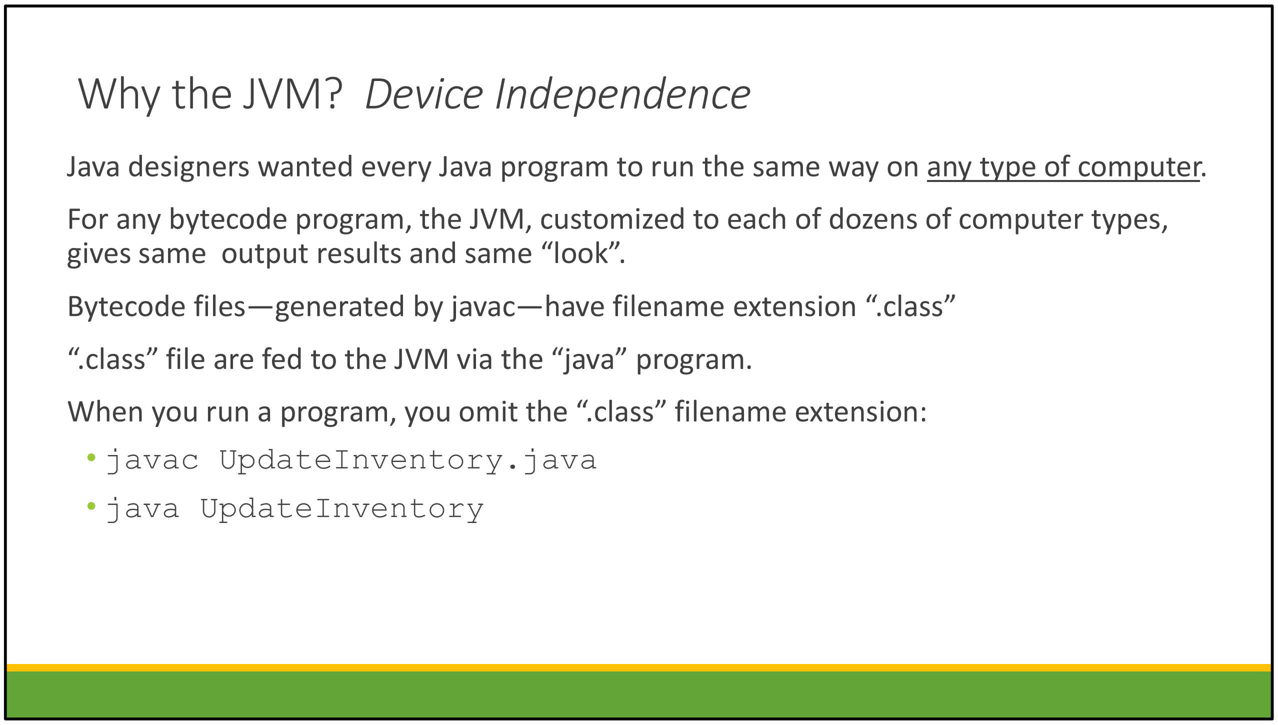
java:
Feeds bytecode to the **J**ava **V**irtual **M**achine
JVM executes bytecode i.e. runs program

What's a Virtual Machine?

The Java Virtual Machine looks like a CPU that has bytecode as its machine language

- JVM is actually software—but "acts like" a CPU





Why the JVM? *Device Independence*

Java designers wanted every Java program to run the same way on <u>any type of computer</u>.

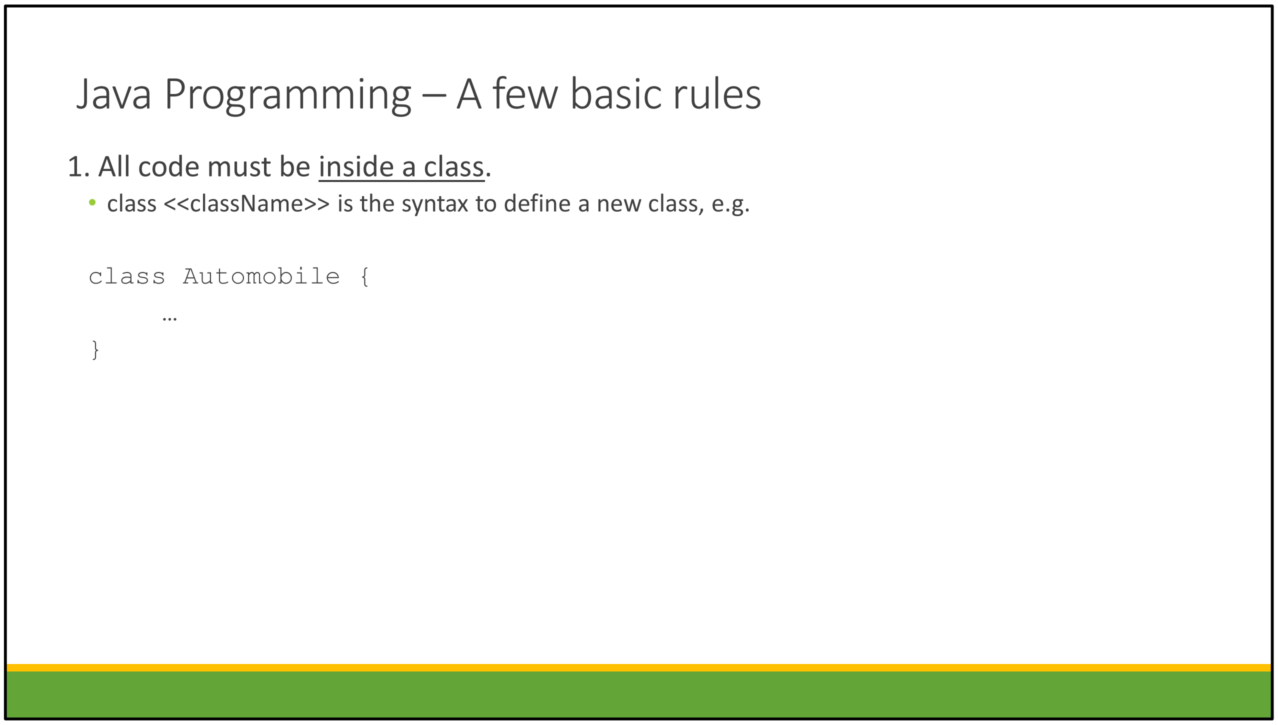
For any bytecode program, the JVM, customized to each of dozens of computer types, gives same output results and same “look”.

Bytecode files—generated by javac—have filename extension “.class”

“.class” file are fed to the JVM via the “java” program.

When you run a program, you omit the “.class” filename extension:

- `javac UpdateInventory.java`
- `java UpdateInventory`



Java Programming – A few basic rules

1. All code must be inside a class.
   - class <<className>> is the syntax to define a new class, e.g.

```
class Automobile {
        …
}
```

And the definition of the class is inside a matching pair of curly braces
- Parentheses, brackets, curly braces

Java Programming – A few basic rules

2. A class contains *methods* and *variables*

- Methods and variables have names, e.g.

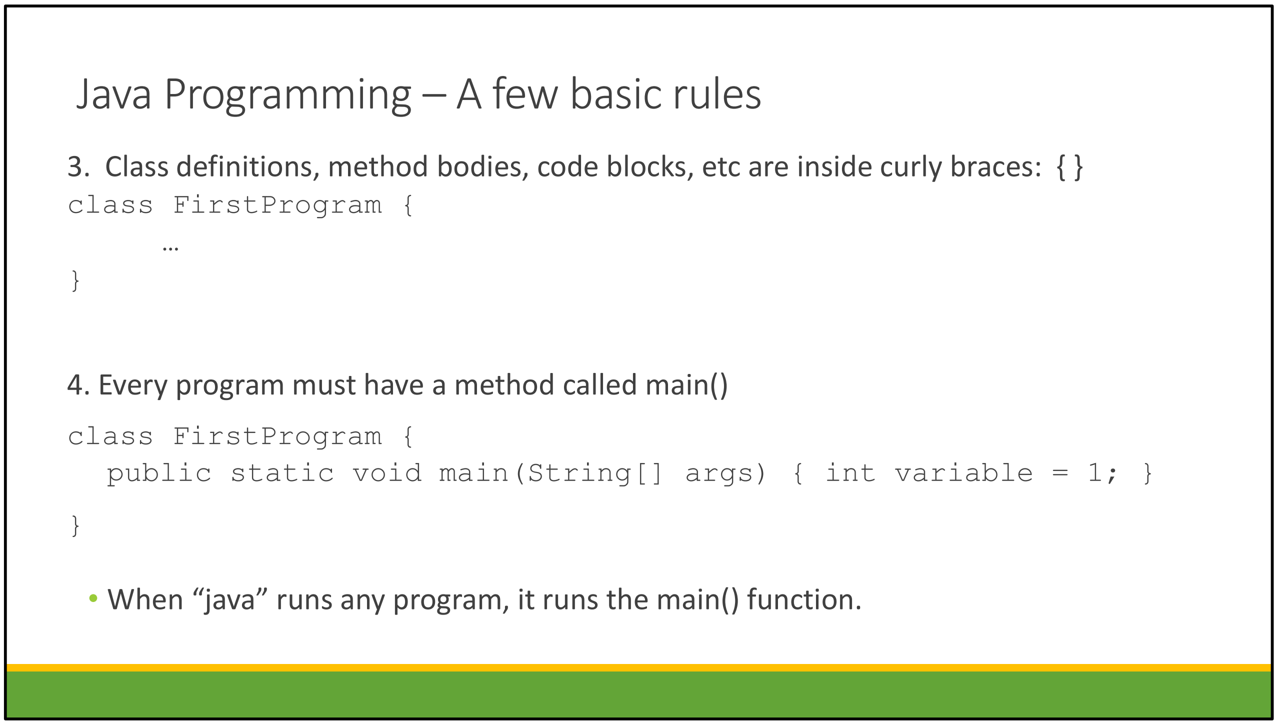
```
int MyStudentId()
```

- Methods contain *statements*
- A statement is a line of code that does something, e.g.

```
myCounter = 10;
```

- Statements end in a semicolon

"Method", "function", and "procedure" all mean the same thing: a block of statements with a name (the "method name") that hopefully does something useful and that can be executed ("called") by other methods.



Java Programming – A few basic rules

3. Class definitions, method bodies, code blocks, etc are inside curly braces: { }

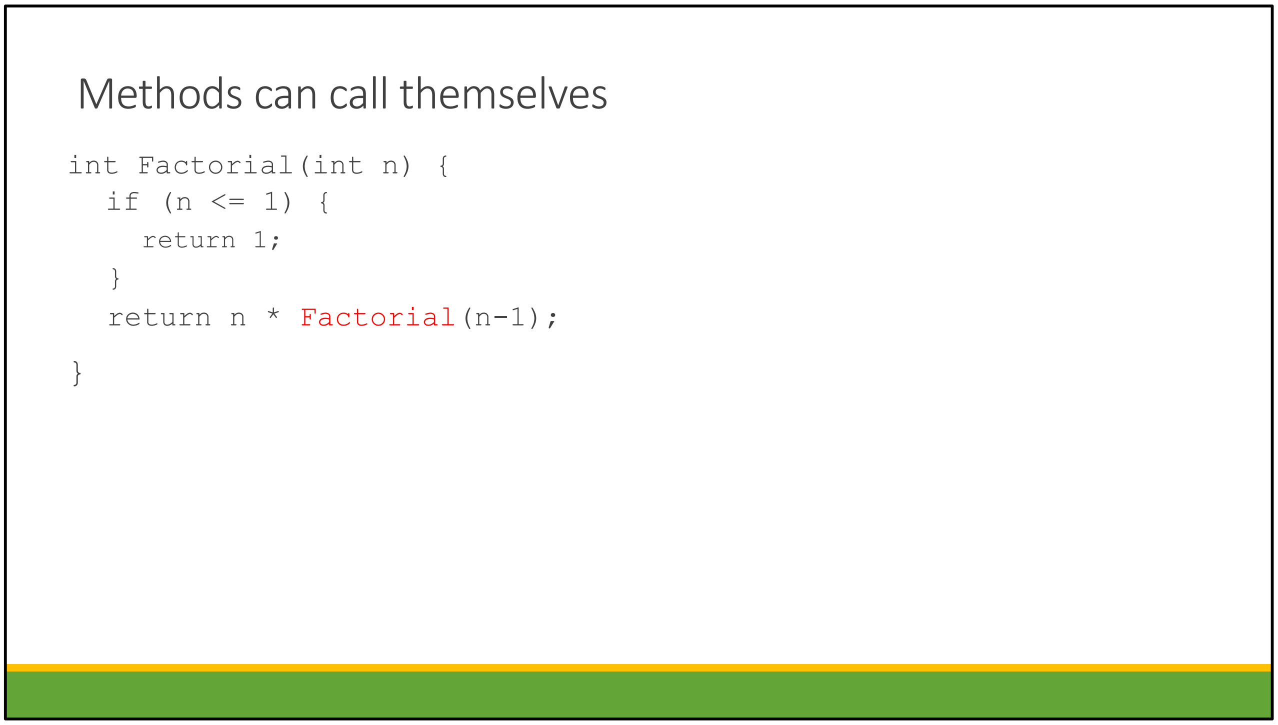
```
class FirstProgram {
      …
}
```

4. Every program must have a method called main()

```
class FirstProgram {
  public static void main(String[] args) { int variable = 1; }
}
```

- When "java" runs any program, it runs the main() function.

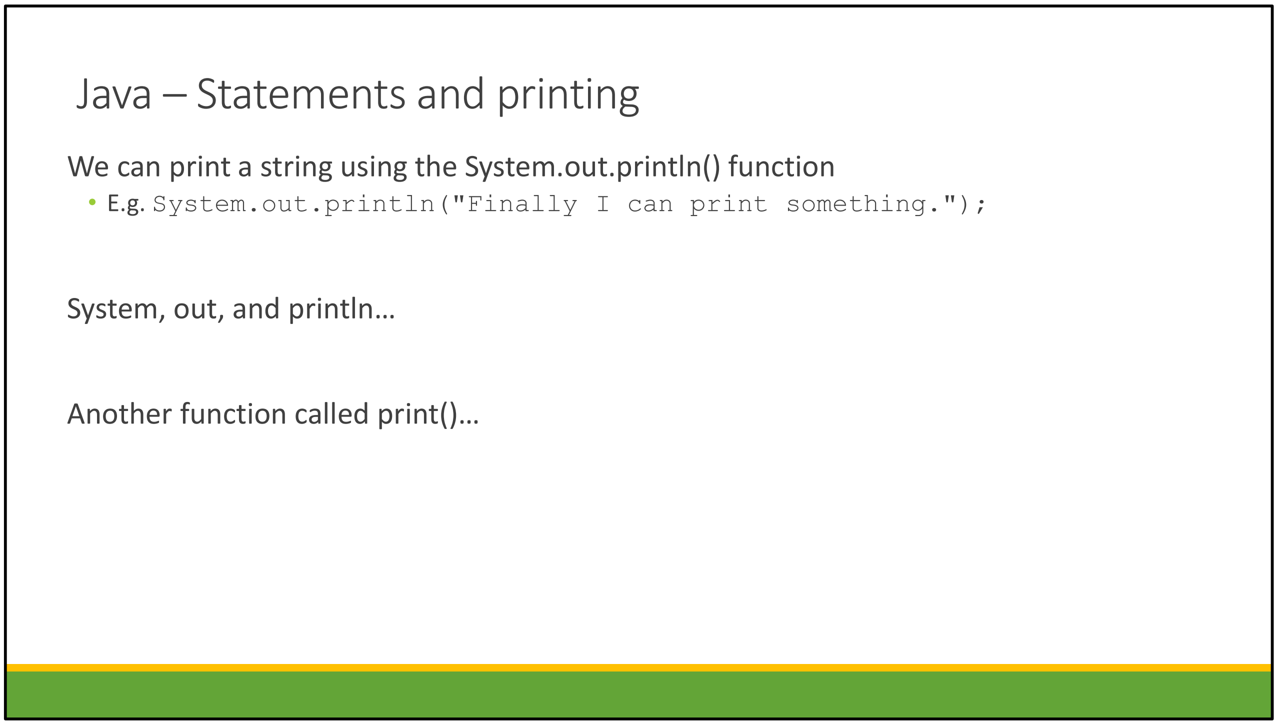
Let's talk about public, static, void, String later. For now, just copy them.



Methods can call themselves

```
int Factorial(int n) {
  if (n <= 1) {
    return 1;
  }
  return n * Factorial(n-1);
}
```

What is Factorial? Explain: Factorial(3) = 3*2*1. Factorial(5) = 5*4*3*2*1
Explain how code works
"RECURSION" – will discuss at length later



Java – Statements and printing

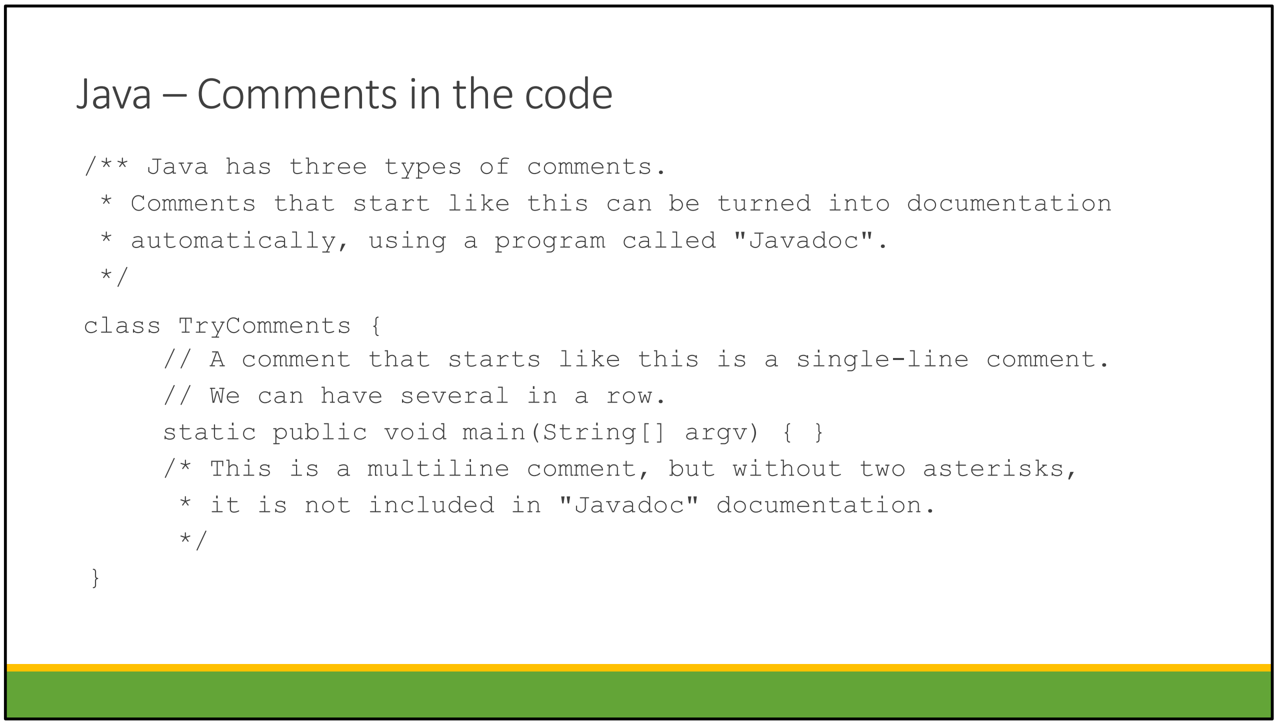
We can print a string using the System.out.println() function

- E.g. `System.out.println("Finally I can print something.");`

System, out, and println…

Another function called print()…

System is a built-in library in Java. It supports lots of system functions like printing, reading keyboard input, etc
out is an object in the System library
Println is a function that belongs to out.
Show program with print() and println() in Eclipse!



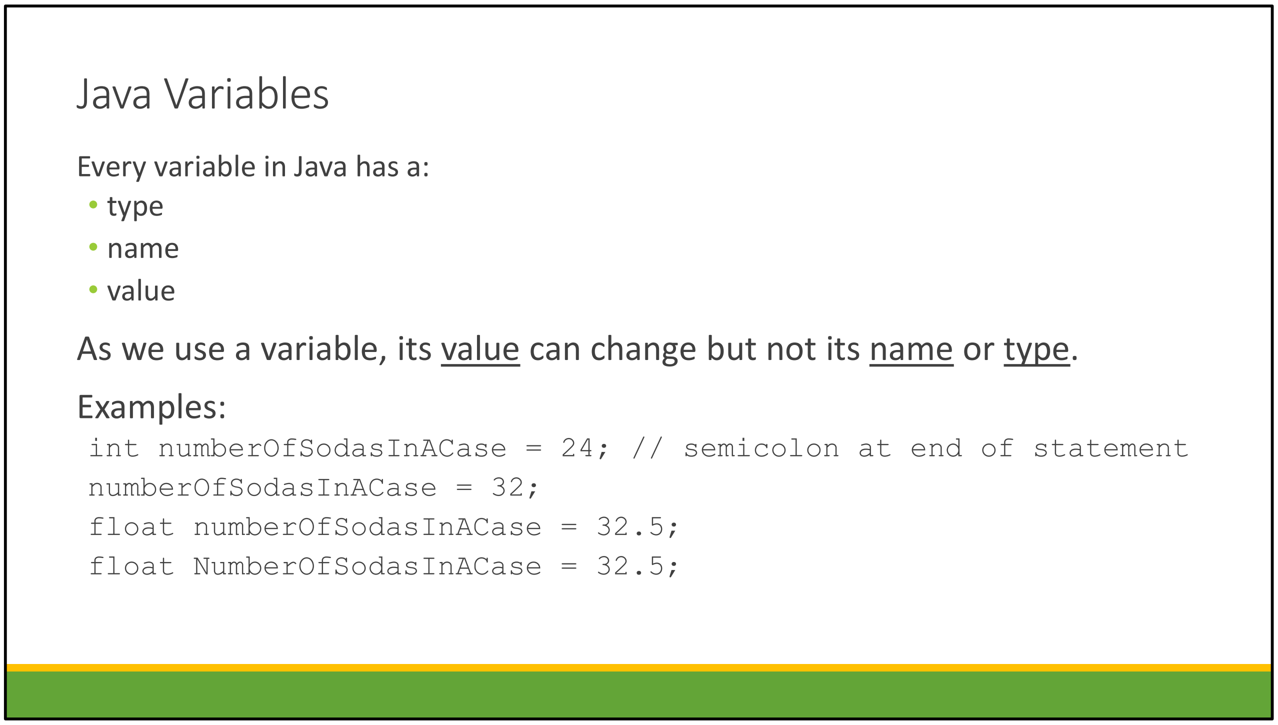
Java – Comments in the code

```
/** Java has three types of comments.
 * Comments that start like this can be turned into documentation
 * automatically, using a program called "Javadoc".
 */
class TryComments {
     // A comment that starts like this is a single-line comment.
     // We can have several in a row.
     static public void main(String[] argv) { }
     /* This is a multiline comment, but without two asterisks,
      * it is not included in "Javadoc" documentation.
      */
}
```

Why do we add comments?
- Help our future selves, and others, understand how program works
- Creates program documentation for users, via Javadoc
- Can keep notes for work that remains to be done

Your programs should have comments, to explain what programs do and how they work.
Especially tricky bits
Don't do dumb comments! int x = 1; // set x to the value 1

Variables in Java



Java Variables

Every variable in Java has a:

- type
- name
- value

As we use a variable, its value can change but not its name or type.

Examples:

```
int numberOfSodasInACase = 24; // semicolon at end of statement
numberOfSodasInACase = 32;
float numberOfSodasInACase = 32.5;
float NumberOfSodasInACase = 32.5;
```

Which line is no good? "float numberOfSodasInACase…". That variable already exists (and has type int)
What is difference from Python? In Python we can change the type of a variable

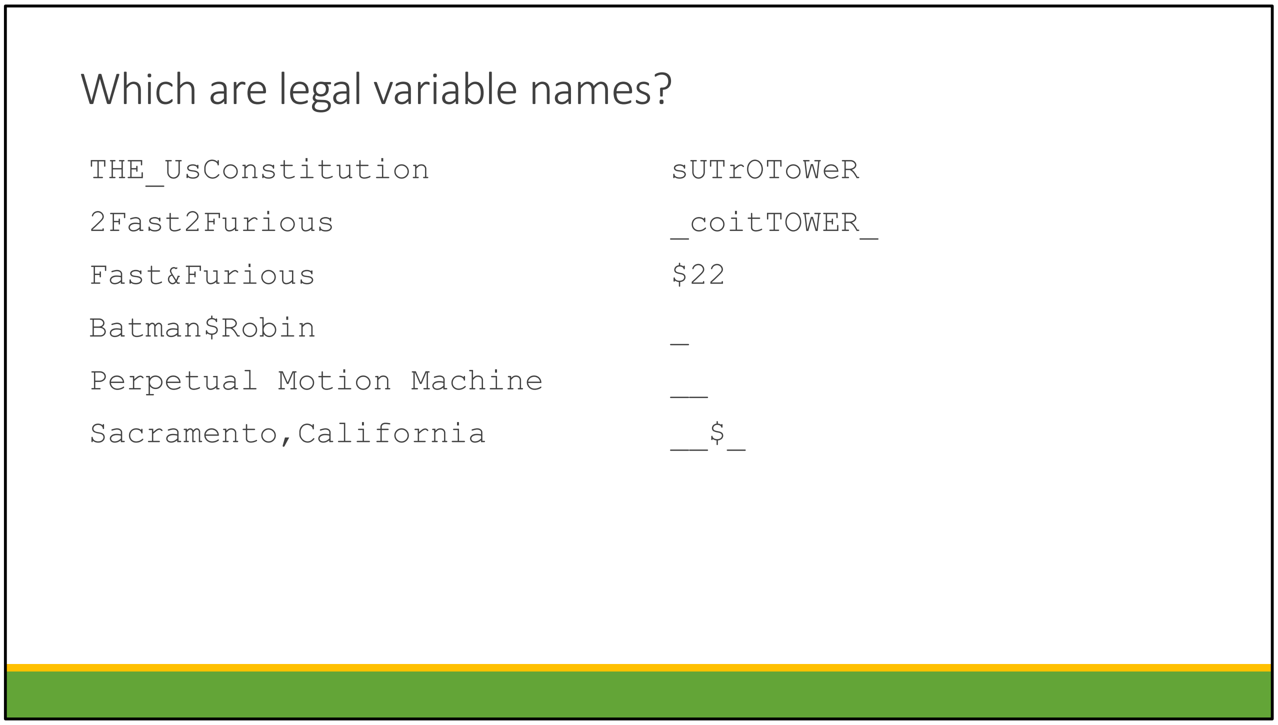
Java Variable Names

A legal variable name in Java:

- May include letters, numerical digits, the underscore '_', and/or the '$' symbol
- Cannot start with a numerical digit
- May not be one of Java's built-in "reserved" keywords e.g. "class", "static", "pubic"
- Letters are case-sensitive

Common style:

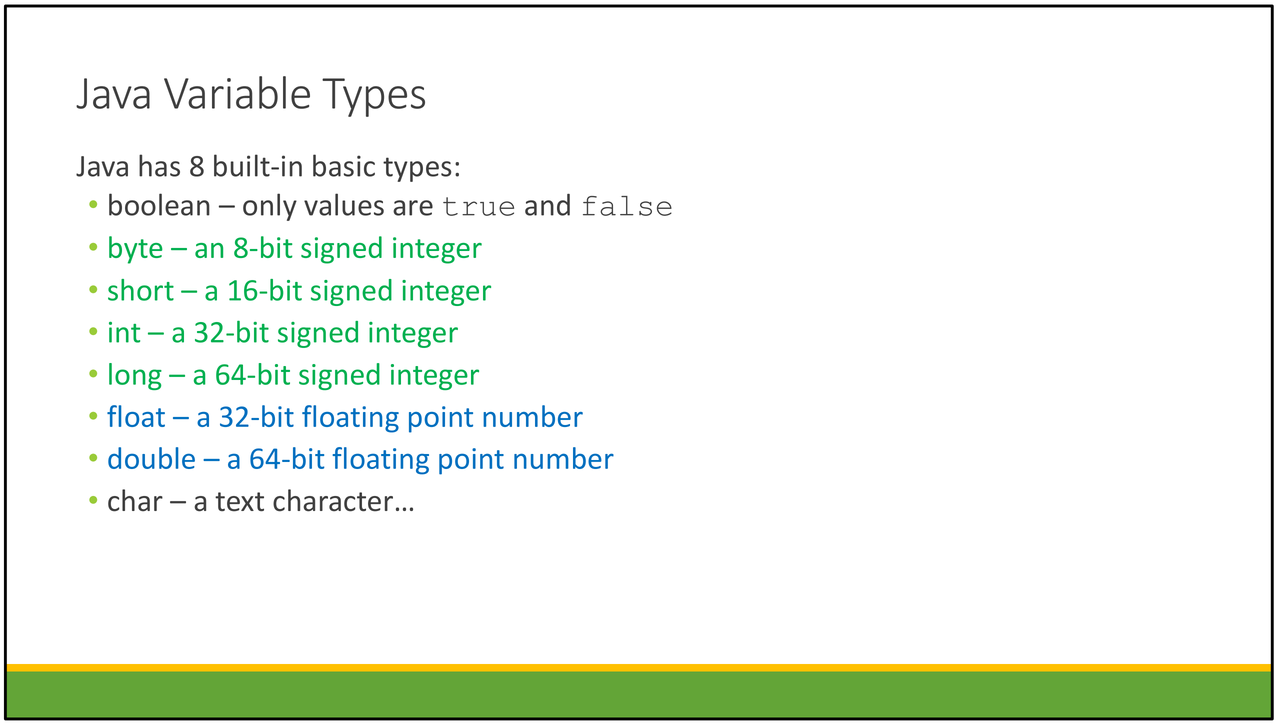
- Start with a letter
- Never use the '$' character
- Use "camel case" for variable names e.g. `allWordsAfterFirstAreCapitalized`
- Use all-caps for constants e.g. `double POUNDS_PER_KILO = 2.2046;`



Which are legal variable names?

```
THE_UsConstitution              sUTrOToWeR
2Fast2Furious                   _coitTOWER_
Fast&Furious                    $22
Batman$Robin                    _
Perpetual Motion Machine        __
Sacramento,California           __$_
```

Note that "_" is a reserved keyword!



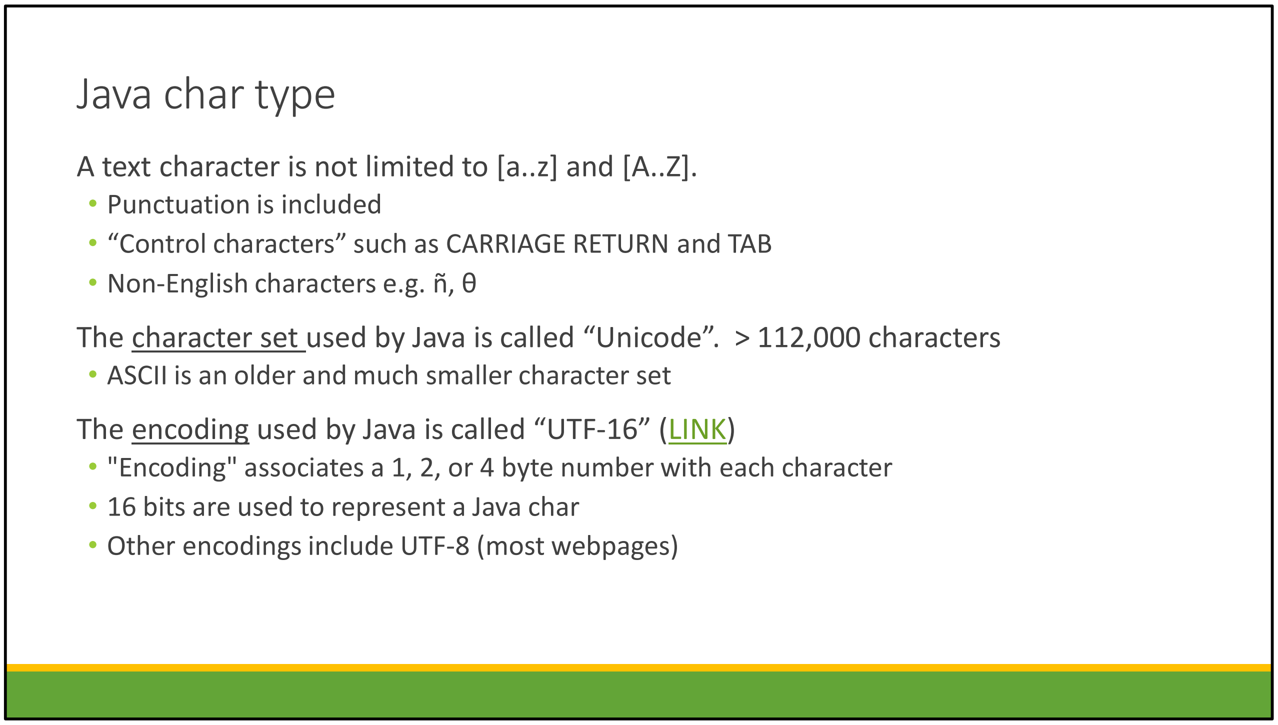
Java Variable Types

Java has 8 built-in basic types:

- boolean – only values are `true` and `false`
- byte – an 8-bit signed integer
- short – a 16-bit signed integer
- int – a 32-bit signed integer
- long – a 64-bit signed integer
- float – a 32-bit floating point number
- double – a 64-bit floating point number
- char – a text character…

Why would anyone ever use anything other than the biggest data type? Speed: of networking, of processing

- Paul's son Scott and his 30 GB data files: too slow to transfer
- Paul's work and processing 32 bytes in a single instruction: faster to process smaller data types



Java char type

A text character is not limited to [a..z] and [A..Z].

- Punctuation is included
- "Control characters" such as CARRIAGE RETURN and TAB
- Non-English characters e.g. ñ, θ

The character set used by Java is called "Unicode". > 112,000 characters

- ASCII is an older and much smaller character set

The encoding used by Java is called "UTF-16" (LINK)

- "Encoding" associates a 1, 2, or 4 byte number with each character
- 16 bits are used to represent a Java char
- Other encodings include UTF-8 (most webpages)

ASCII HAS 128 CHARACTERS. Extended ASCII HAS 256
"UTF" = Unicode Transformation Format
Kinda interesting that WWW and Java use different encodings. Java does have libraries to translate between…
CLICK THE LINK!

Java char type - initialization

A char variable can be initialized with a quoted text character e.g.

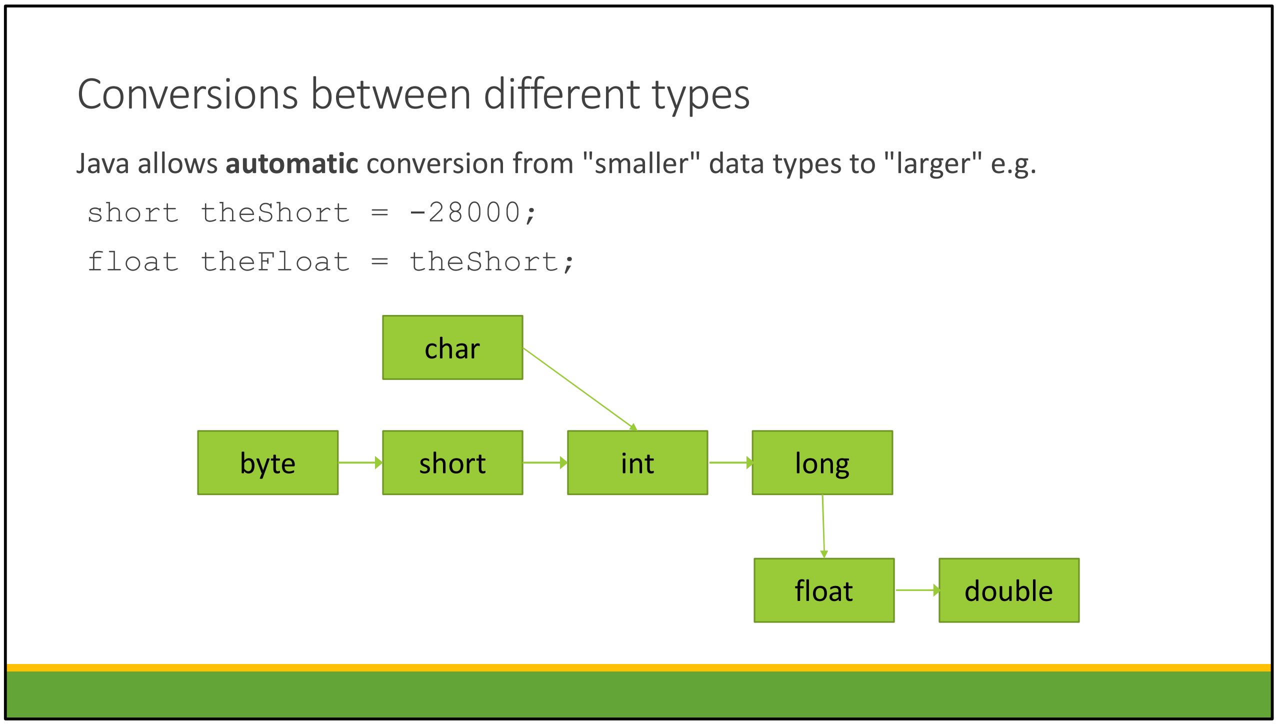
```
    char v1 = 'Y';
```

A char variable can be initialized with an integer constant e.g.

```
    char v2 = 97; // character 'a'
```

A char variable also can be initialized with a “UTF-16” code e.g.

```
    char v3 = '\u0990'; // ঐ (Bengali)
    ("Backslash-u" notation)
```



Conversions between different types

Java allows **automatic** conversion from "smaller" data types to "larger" e.g.

```
short theShort = -28000;
float theFloat = theShort;
```

$$\begin{tikzcd}
 & \text{char} \arrow[dr] & & & \\
\text{byte} \arrow[r] & \text{short} \arrow[r] & \text{int} \arrow[r] & \text{long} \arrow[d] & \\
 & & & \text{float} \arrow[r] & \text{double}
\end{tikzcd}$$

Any "multi-hop" conversions are automatic also, e.g. byte to double

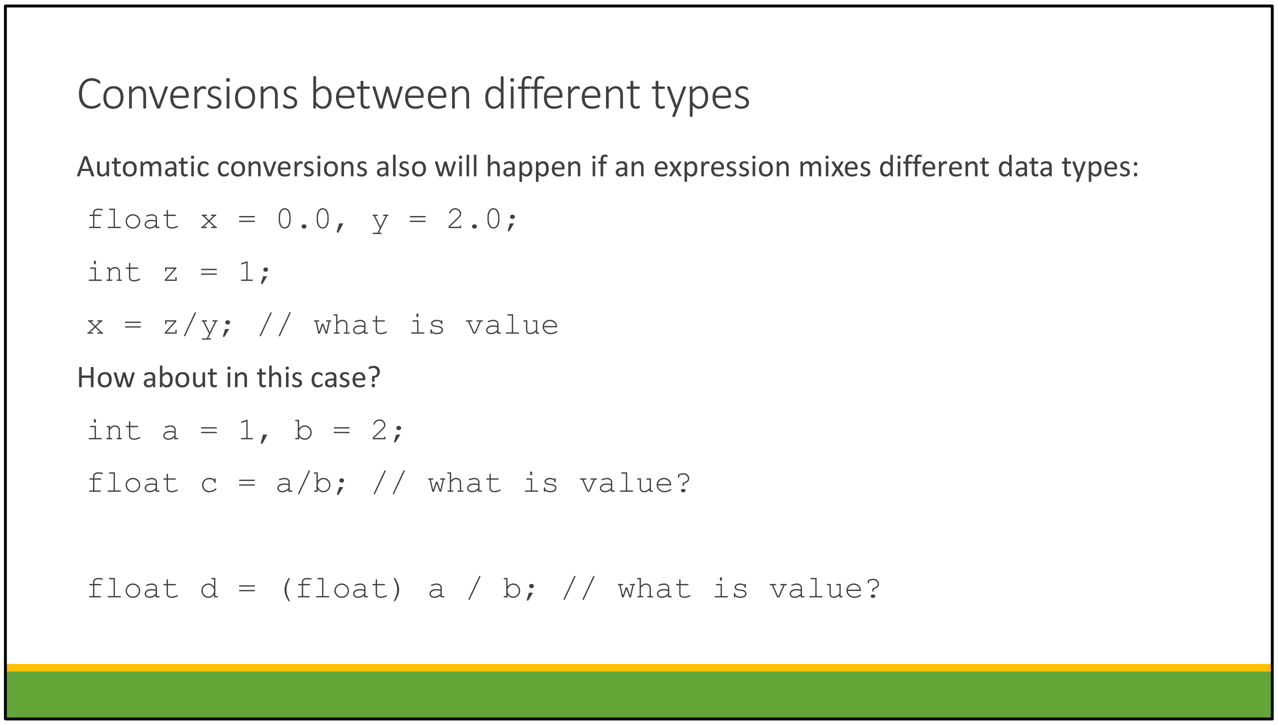
Conversions between different types

When converting from "larger" to "smaller" there is a risk of losing data, e.g:

```
float x = 2.5;
int y = x; // lose the ".5" part
```

Java requires you to use a "cast" to make large-to-small conversions:
put the name of the desired type, in parentheses, before the value you want to convert, e.g:

```
float x = 2.5;
int y = (int) x; // "(int)" is the cast
```



Conversions between different types

Automatic conversions also will happen if an expression mixes different data types:

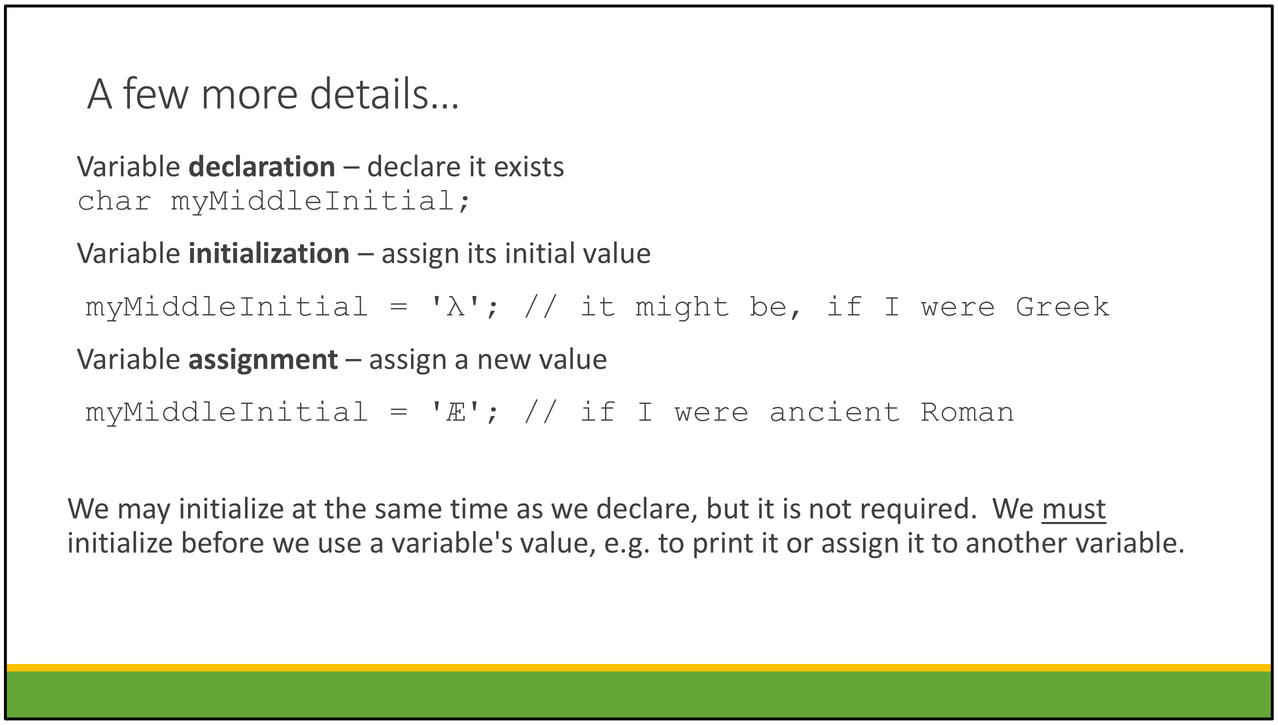
```
float x = 0.0, y = 2.0;
int z = 1;
x = z/y; // what is value
```

How about in this case?

```
int a = 1, b = 2;
float c = a/b; // what is value?

float d = (float) a / b; // what is value?
```

Do this in Eclipse!!!
Top answer is 0.5
Second answer is 0: integer division happens first, get result of 0, assign 0 to 'c'



A few more details…

Variable **declaration** – declare it exists

```
char myMiddleInitial;
```

Variable **initialization** – assign its initial value

```
myMiddleInitial = 'λ'; // it might be, if I were Greek
```

Variable **assignment** – assign a new value

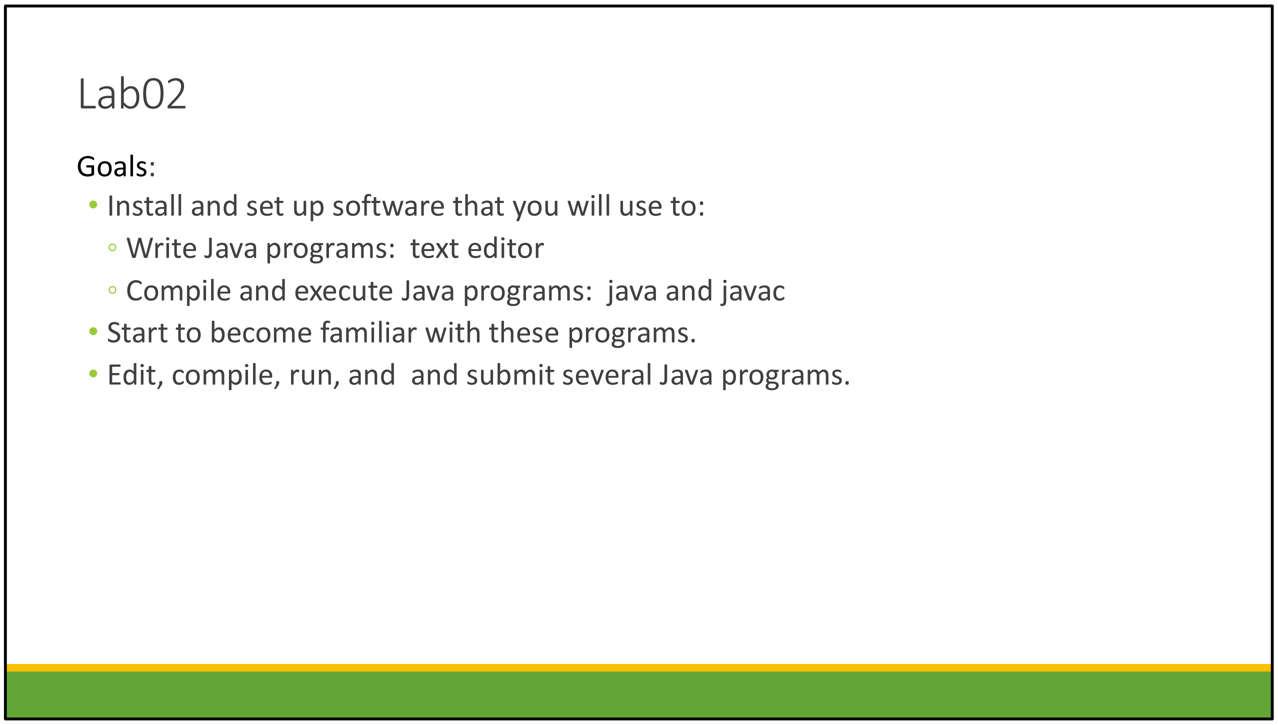
```
myMiddleInitial = 'Æ'; // if I were ancient Roman
```

We may initialize at the same time as we declare, but it is not required. We must initialize before we use a variable's value, e.g. to print it or assign it to another variable.

WHY MIGHT WE like TYPED VARIABLES, UNLIKE IN PYTHON?
- Compiler catches errors. We don't have to find them thru testing

Lab 02

Let's do some Java



Lab02

Goals:

- Install and set up software that you will use to:
  - Write Java programs: text editor
  - Compile and execute Java programs: java and javac
- Start to become familiar with these programs.
- Edit, compile, run, and and submit several Java programs.

Java Installation

There are several Java "packages" available. Most common are:

- **J**ava **R**untime **E**nvironment: for Java users. Includes "java" program, JVM, other files such as font files
- **J**ava **D**evelopment **K**it: for Java developers. Includes JRE, plus "javac", documentation tools, debugging tools, etc. <u>We want this one</u>

Use "git pull" in the CourseInfo directory. Look for the **Lab02.pdf** file, read it, and get started!